CLAIMS:

1. A method for checking whether a programmable logic device functions properly when programmed, comprising:

obtaining an integrated circuit design for the programmable logic device, the integrated circuit design including at least a programmable logic portion of the programmable logic device;

obtaining at least one test pattern to test at least the programmable logic portion;

obtaining memory states for configuring the programmable logic device;

applying the memory states to at least the programmable logic portion to configure the programmable logic portion with the at least one test pattern to provide a configured design; and

equivalency checking the configured design with the at least one test pattern.

2. A method as in claim 1, wherein the step of applying comprises:

filtering the memory states responsive to the programmable logic portion; and

hierarchically combining logic blocks for the at least one test pattern as configured to provide the configured design.

3. A method as in claim 1, wherein the step of applying comprises:

respectively configuring the at least one test pattern into at least one sub-block of the programmable logic portion to provide the configured design; and

storing at least one of the test pattern as configured.

4. A method as in claim 1, wherein the step of equivalency checking is done at a Boolean level.

5. A method for checking whether a programmable logic device functions properly when configured, comprising:

obtaining an integrated circuit design for the programmable logic device, the integrated circuit design represented at a first level of abstraction;

obtaining a test case design, the test case design including a plurality of individual circuits, the test case design represented at the first level of abstraction;

synthesizing each of the plurality of individual circuits to provide network nodal state information for each of the plurality of individual circuits;

obtaining memory states for each of the plurality of individual circuits from the network nodal state information;

applying the memory state statements to the integrated circuit design;

level abstracting the integrated circuit design with the memory state statements applied to provide configured circuits represented at a second level of abstraction; and

equivalency checking the configured circuits with the plurality of individual circuits.

6. A method as in claim 5, wherein the first level of abstraction is lower than the second level of abstraction.

7. A method as in claim 6, wherein the second level of abstraction is selected from gate level and register transfer level.

8. A method as in claim 7, wherein the first level is at or below a transistor level.

9. A method as in claim 5, wherein the integrated circuit design is for a programmable logic device.

10. A method as in claim 9, wherein the programmable logic device is a field programmable gate array.

11. A method as in claim 5, wherein the step of level abstracting comprises:

filtering the memory state statements responsive to the plurality of individual circuits; and

hierarchically combining the plurality of individual circuits as respectively configured with the memory state statements to provide the configured circuits as a configured design.

12. A method as in claim 11, wherein the step of level abstracting comprises:

respectively configuring each of the plurality of individual circuits into unconfigured programmable fabric of the integrated circuit design; and

storing each of the plurality of individual circuits as configured.

13. An apparatus for checking whether a programmable logic device functions properly when configured, comprising:

database means containing an unconfigured design for the programmable logic device and test patterns;

a placer and router for placing and routing the test patterns to provide network nodal information for the test patterns;

means for obtaining memory states for the test patterns from the network nodal information;

means for applying the memory states for the test patterns to the unconfigured design and for configuring a portion of the unconfigured design with the test patterns to

provide a configured design; and

an equivalency checker to compare the test patterns with the configured design.

14. An apparatus as in claim 13, wherein the means for applying comprises a level abstractor configured to hierarchically assemble the test patterns.

15. An apparatus as in claim 14, wherein the means for obtaining the memory states comprises a map file input and translation to a target format.

16. An apparatus as in claim 15, wherein the means for obtaining the memory states comprises a bitstream input.

17. An apparatus as in claim 13, wherein the equivalency checker is configurable to operate at a level of abstraction selected from transistor, gate, register transfer, behavioral and system levels.

18. A signal-bearing medium having a program, which when executed with a programmed computer, causes execution of a method for checking whether a programmable logic device functions properly when configured, comprising:

accessing a first database for obtaining an unconfigured design for the programmable logic device;

accessing a second database for obtaining test patterns;

accessing a suite of software tools to obtain memory states for the test patterns;

applying the memory states for the test patterns to the unconfigured design to configure at least a programmable logic portion of the unconfigured design with the test patterns to provide a configured design; and

accessing the suite of software tools to equivalency check the test patterns with the configured design.

19. A system for testing whether a programmable logic device functions properly when configured, comprising:

a computer including at least one processor, at least one input/output interface, and memory;

mass storage memory accessible by the computer to store a suite of software tools and databases;

the memory having a program which when executed by the computer causes,

a first database of the databases to be accessed to obtain an unprogrammed design for the programmable logic device;

a second database of the databases to be accessed to obtain test patterns;

a tool in the suite of software tools to obtain memory states for the test patterns;

the memory states for the test patterns to be applied to the unconfigured design to configure a programmable logic portion of the unconfigured design with the test patterns to provide a configured design; and

another tool in the suite of software tools to equivalency check the test patterns against the configured design to determine if the configured design is functionally equivalent to the test patterns.

20. A method for checking whether a programmable logic device functions properly when configured, comprising:

obtaining an integrated circuit design for the programmable logic device;

obtaining a test case design;

synthesizing the test case design;

obtaining memory states from the test case design synthesis;

obtaining a first logic block from the integrated

circuit design;

configuring the first logic block with a first portion of the test case design;

applying first information from the memory states to the logic block;

obtaining a second logic block from the integrated circuit design;

configuring the second logic block with a second portion of the test case design;

applying second information from the memory states to the second logic block;

hierarchically combining the first logic block and the second logic block to provide a design abstraction; and

equivalency checking the design abstraction with the test case design.

21. A method as in claim 20, wherein the steps of configuring the first logic block and applying the first information are done with level abstraction.

22. A method as in claim 20, wherein the step of configuring the first logic block is done with level abstraction, and wherein the step of applying the first information is done with logic synthesis.

23. A method as in claim 22, wherein the logic synthesis is done prior to the level abstraction.

24. A method as in claim 22, wherein the logic synthesis is done after the level abstraction.

25. A method for checking whether a programmable logic device functions properly when configured, comprising:

obtaining a logic block;

obtaining memory states;

filtering the memory states responsive to the logic block to provide filtered memory states;

translating the filtered memory states to the logic block;

parsing out a cell of the logic block from a hierarchical relationship with another cell; and

renaming the cell of the logic block.

26. A method as in claim 25, further comprising:

obtaining another logic block;

filtering the memory states responsive to the other logic block to provide other filtered memory states;

translating the other filtered memory states to the logic block;

parsing out the other cell from the hierarchical relationship; and

renaming the other cell of the other logic block.

27. A method as in claim 26, further comprising:

combining the cell as renamed and the other cell as renamed into the hierarchical relationship.

28. A method as in claim 27, wherein the translating the filtered memory states to the logic block comprises abstracting the logic block to a register-transfer-level model.

29. A method as in claim 28, wherein the combining into the hierarchical relationship comprises re-abstracting the register-transfer-level model with a top-level netlist to provide a top-level register-transfer-level model.

30. A method as in claim 29, wherein the logic block is described in Verilog including at least one tranif Verilog statement.

31. A method as in claim 30, wherein the re-abstracting comprises using the at least one tranif Verilog statement to provide for bi-directional port-to-port signaling having no logical operations.